Max-product algorithm and its simplification

In this section we give a brief introduction of the max-product algorithm applied in our model inference procedures. Max-product is a standard belief propagation algorithm on factor graph models. A comprehensive tutorial can be found in in the paper "Factor graphs and the sum-rpoduct algorithm" by Kschischang, Frey and Loeliger on IEEE Transactions on Information Theory. We also discuss the methods of simplifying the maximization of the message passing step. A detailed description will be provided in a longer paper.

Define a *message* as a function associated with an edge in the factor graph. It takes the variable node of the corresponding edge as the argument. For instance, $m_{\phi_1 \to x_1}(x_1)$ is a message from ϕ_1 to x_1 and is a function of x_1. Notice messages are directed, hence $m_{\phi_1 \to x_1}(x_1)$ and $m_{x_1 \to \phi_1}(x_1)$ can be different. A message must be either from a variable to a factor or vice versa for the bi-partite property of the factor graph.

Messages can be propagated to adjacent nodes and updated according to the messages of adjacent nodes. There are two rules of updating messages. An updated message from a variable x to a factor f is simply the product of all messages incident to x except from f:

$$m_{x \to f}(x) = \prod_{f_i \in N(x) \backslash \{f\}} m_{f_i \to x}(x). \tag{1}$$

where $N(x)$ denotes the neighboring factor nodes of x. In contrast, an updated message from a factor f to a variable x is the maximization of the product of the factor function and incident messages over configurations in $N(f) \backslash \{x\}$:

$$m_{f \to x}(x) = \max_{N(f) \backslash \{x\}} f(x, N(f) \backslash \{x\}) \prod_{x_i \in N(f) \backslash \{x\}} m_{x_i \to f}(x_i). \tag{2}$$

The max-product algorithm executes simultaneous updates of all messages until all messages converge to fixed functions. With proper initialization, it can compute either unconditional or conditional max-marginal probabilities.

Figure 1: Max-product algorithm

1. Initialize message. An initial message from a variable node x is $m^0_{x \to f}(x) = 1$ if x is not conditioned to a fixed value, and $m^0_{x \to f}(x) = I(x = c)$ if x is fixed to value c. An initial message from a factor node f is $m^0_{f \to x}(x) = \sum_{N(f) \backslash \{x\}} f(x, N(f) \backslash \{x\})$.
2. Iteratively pass messages to their neighbors (except the nodes where they come from) and update messages according to equations 1 and 2.
3. Terminate when all messages converge to fixed functions.
4. The belief function of a variable x is the product of its incident messages:

$$b(x) = \prod_{f_i \in N(x)} m_{f_i \to x}(x). \tag{3}$$

The belief functions are the exact max-marginal probabilities when the factor graph contains no loops. Otherwise the inferred beliefs are approximations, or the messages may even fail to converge. The quality of the belief propagation approximation and the development of better approximate (or exact) inference algorithms on loopy graphs are currently under intensive study. In our work, the quality of the inference results does not

seem to hinder the model, for the inferred configurations are consistent with all constraints in the analysis of mating response pathways.

One possible bottleneck in the max-product algorithm is the evaluation of factor $\rightarrow$ variable messages (equation 2). It is inefficient to calculate $m_{f\rightarrow x}(x)$ by enumerating all configurations in the arugments of f. The structure of the potential functions in our setting, however, permits efficient evaluation of these messages. If $\psi_f(.)$ is a potential function of a physical data or a knock-out observation, then $m_{f\rightarrow x}(x)$ can be efficiently calculated because $\psi_f(.)$ contains only one variable. If $\psi_f(.)$ is a potential function of knock-out explanation, then we can calculate the upper bound of certain configurations without visiting them individually. For example, suppose we want to evaluate a message $m_{f\rightarrow x}(x)$ from a single path factor f to an edge presence variable x. For $x = 0$, the best scenario is either this path is not selected ($\sigma = 0$), the knock-out effect is insignificant ($k_{ij} = 0$), or both conditions hold. The max configurations of other varables in each scenario are determined by their incident messages but are not dependent on the potential function. Hence the max configuration for $x = 0$ is the supremum among the three cases. For $x = 1$ we need to consider two scenarios: either the path is not selected, or the path is selected and all other variables satisfy the constraints of explanation. The best scenario for the path is not selected is the same as $x = 0$ case. For the path is selected and the knock-out effect is explained, all variables except edge signs are fixed. Since the path length is restricted, we can enumerate all edge sign configurations to obtain the best scenario. The max configuration for $x = 1$ is the supremum of these two cases. The message incident from a noisy OR function of path selections can be calculated analogously. This simple deduction greatly reduces the number of enumerations to consider. Moreover, instead of storing the whole lookup table we only need to store distinct return values, since the configurations leading to a particular return value can be deduced from the constraint. This simplification applies to all potential functions composed of simple logical rules such as path explanation and noisy-OR.

Recursive algorithm of model decomposition

As demonstrated in the paper, an optimal configuration in the physical network model can be expressed as a product of configurations in subnetworks. Variables in different subnetworks vary independently given the constraints from the current data. In this section we describe a recursive algorithm to identify these subnetworks.

The variables which are invariant across all MAP configurations have unique max-marginal probabilities. They are obtained by running the max-product algorithm once. We then continue running the max-product algorithm by conditioning on the evidence obtained from the previous step. The variables with distinct conditional max marginal probabilities are fixed according to their arg max values of conditional max marginal probabilities. In each step, we also choose a variable with degenerate conditional max marginal probabilities and set it to one of the arg max values. This process continues until all variables are fixed. Clearly, variables which are fixed after running max-product depend on the variables which are externally set at the same step. Moreover, the variables which are externally set at earlier steps can also affect the variables which are fixed later. These dependence relations can be established by verifying whether the two variables appear in the same potential function (meaning that they are constrained together). Variables which are dependent belong to the same subnetwork.

The algorithm of decomposing subsystems is described as follows.

Figure 2: Recursive algorithm for decomposing MAP configurations

1. Find variables $X_I = \{x_i : i \in I\}$ such that $P^{max}(x_i)$ favors a unique max value at $\hat{x}_i$ for each x_i. $X_I = (\hat{x}_i : i \in I) = \hat{X}_I$ is the invariant subconfiguration of the model.
2. Recurse on the following steps until all variables are fixed.
   (a) Select an x_i which is not yet fixed.
   (b) Find one optimal value of x_i denoted by v_i. Set $x_i = v_i$.
   (c) Run max-product algorithm to compute $P^{max}(x|x_i = v_i)$.
   (d) $\forall x$ which is not yet fixed and $\hat{v} = \arg\max_v P^{max}(x = v|x_i = v_i)$ is unique, establish $x_i \succ x$.
   (e) $\forall x$ which satisfies previous conditions, $\forall x_j$ which has been externally set at previous steps. If $\exists \phi_l(x, x_j, .)$ such that $m_{\phi_l \to x}(x)$ has the same max value as $P^{max}(x|x_i = v_i)$, then $x_j \succ x$.
   (f) Recurse to 2.
3. Construct a graph G_d of degenerate variables induced by binary relation $\succ$.
4. Identify connected components in G_d. The submodels $V_1, \cdots, V_k$ are vertex sets of connected components.